ERDC MSRC Major Shared Resource Center

ERDC MSRC/PET TR/00-25

Tools for Handheld Supercomputing: an Assessment of the Wireless Application Protocol (WAP)

by

David E. Bernholdt
Sangyoon Oh
Konrad Olszewski
Geoffrey C. Fox

30 May 2000

07h0062000

Tools for Handheld Supercomputing: an Assessment of the Wireless Application Protocol (WAP)

David E. Bernholdt, Sangyoon Oh,
Konrad Olszewski, Geoffrey C. Fox
{bernhold,soh,konrad,gcf}@npac.syr.edu
Northeast Parallel Architectures Center
Syracuse University
Syracuse, NY 13244-4100

Abstract

The Wireless Applications Protocol (WAP) is an emerging industry standard approach to the delivery of web-like services to small mobile digital devices. Designed primarily with the booming cellular phone market in mind, WAP specifically takes into account the limited display, memory, input, and performance capabilities typical of such handheld devices, as well as the unreliability and limited bandwidth and high latency often associated with wireless data communications technologies. We describe the relevant aspects of the WAP standard, compare WAP with more familiar technologies widely deployed in the WWW domain, and try to understand the role that WAP is likely to play in the development of "handheld supercomputing" applications which link mobile digital devices to high-performance computers.

1 Introduction

High-performance computing has a long and illustrious history. Such machines are indispensable in providing the kind of resources and performance that are required for modern computational science and engineering simulation applications. Today, users typically access these systems over local-area and wide-area networks from personal computers (desktop workstations, laptops, etc.). At the same time, we are seeing a tremendous growth in the variety and capability of handheld digital devices and in wireless communications connectivity. As digital mobile phone service becomes more widespread and prices decrease, it is becoming more practical to develop products which couple modest (but non-trivial) computing capabilities with wireless data connectivity.

Cellular phones and personal digital assistants (PDAs) are far from the only examples of these changes, but they are particularly relevant in the context of "handheld supercomputing." PDAs, which run operating systems such as Palm OS, Windows CE, and EPOC have achieved a form factor and functionality which makes them extremely popular. They also provide a level of user programmability that has not been available in previous generations of PDAs, allowing a large market of third-party software to develop, and allowing the development of highly specialized and personalized applications. Current generation PDAs offer low power-consumption CPUs and memory typically ranging from 1-32 MB which give them modest but useful computing power. I/O in these systems is somewhat limited. A small display, perhaps 160x160 pixels, pen-based input or micro-keyboard, a serial port, and simple audio capabilities are typical; some devices also offer infrared ports and Compact Flash slots. Data on the PDA is usually shared with the user's personal computer system, providing convenient desktop access to the data and backup. The serial port (or IR port) is used to connect the PDA to the computer for synchronization and backup of data. Many vendors provide modems that can be used to provide dialup connectivity, and those with Compact Flash slots can also provide connectivity to local-area ethernets. Wireless connectivity is also starting to appear in the PDA market. The Palm VII from Palm Computing includes a built-in wireless modem that provides CDPD (cellular digital packet data) access to the Internet via Palm Computing's Palm.net gateway; Novatel provides add-on CDPD modems for other Palm models. GSM cellular modems are available in the Compact Flash form factor (though in the United States, GSM coverage is currently more limited than other digital cellular bearers).

With the introduction of digital cell phones, it became easier to mix voice and data applications. At the same time the amount of processing capability in the phone has increased significantly, routinely providing phonebook, messaging, and other services in the handset. Modern cell phones typically have small LCD displays and the numeric keypad (often including some additional buttons and softkeys) for input. They are not user programmable, but a number of providers are starting to offer Internet access services via the handset display. These trends have provided the impetus for the development of the Wireless Applications Protocol (WAP) which will be described in detail below.

The rapid development of both wireless connectivity and handheld digital devices has been driven by consumer markets, but of course use of these technologies are not limited to "commodity" applications. Just as the worldwide web (WWW) has provided an environment that has been extended beyond commodity applications to providing new ways to access HPC-related resources, educational material, and other resources that fall well outside the "commodity" designation. The confluence of mobile computing and wireless connectivity will undoubtedly have a similar result and one can imagine these devices providing capabilities such as job monitoring and submission, simple computational steering as well as other functionality that hasn't even been invented yet. (Such applications we describe with the provocative term "handheld supercomputing.") The Wireless Application Protocol is designed to facilitate the construction of mobile information services and applications, and at a glance would appear to be an important part of the toolbox of anyone developing handheld supercomputing applications. In this report we describe WAP, compare it with components of the more familiar WWW domain, and understand the role it can play in handheld supercomputing.

2 The Origins of WAP

At the beginning of 1998, Global Mobile reported that there were over 200 million wireless subscribers in the world. According to the Strategis Group, there will be over 530 million wireless subscribers by the year 2001. Figure 1 shows the explosive growth of the wireless market. [PHN01]

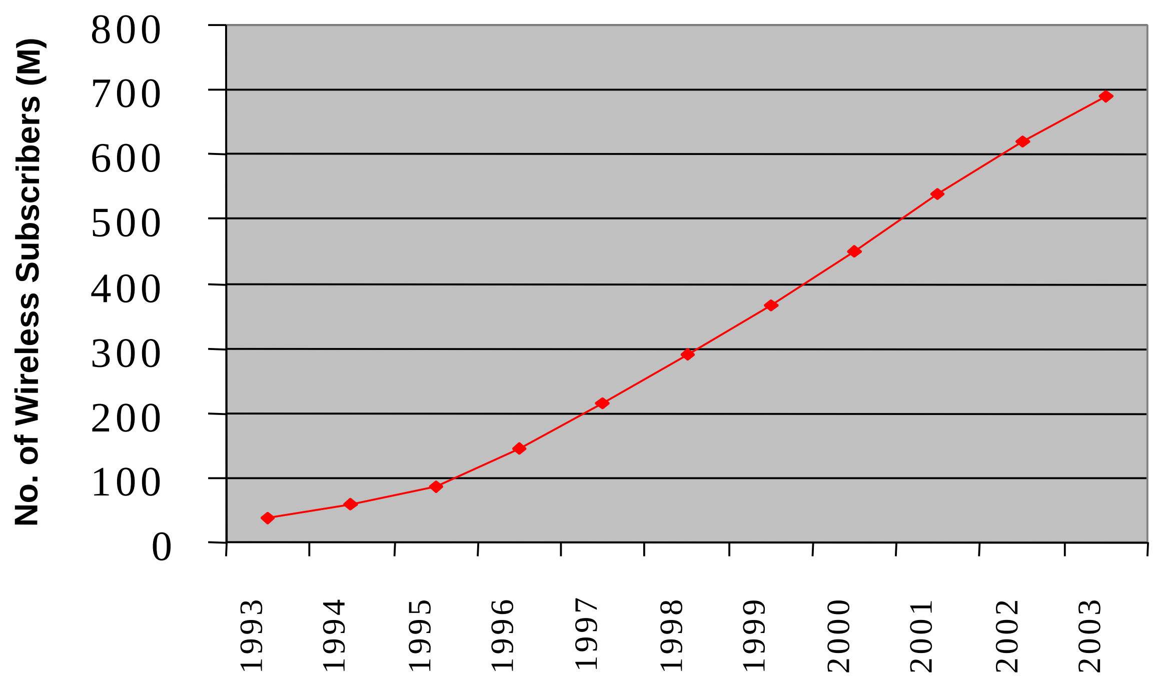

Figure 1: Global Wireless Subscriber Population

Nokia predicted that with this growth would come multimedia capabilities, such as sending and receiving e-mail and push/pull information from the Internet. Extending this trend, wireless devices can be thought of as network appliances, which provide multimedia abilities to play interactive games and to transact e-commerce in addition to e-mail and WWW access. However, despite these trends, subscription rates to wireless *data* lag significantly behind voice usage. One of the reasons for this is that there has been no uniform environment in which to develop such services, greatly increasing the cost of such efforts. Prior to 1997, several mobile phone companies (such as, Ericsson, Nokia and software company Phone.com, at the time known as Unwired Planet) tried to establish general wireless protocols, which they expected to drive future wireless communication network services. Recognizing their mutual interests, the three companies together with Motorola banded together to found the Wireless Applications Protocol Forum developed to provide a framework in which to develop data and multimedia services targeted primarily at mobile phone handsets. In September 1997 the WAP Forum published the basic WAP architecture and in 1998 the first formal specification, WAP 1.0 [PHN02], was released. The WAP Forum now encompasses approximately 360 companies and WAP-enabled devices and software are beginning to appear on the market.

3 Description of the Wireless Application Protocol (WAP)

3.1 Overview

The WAP Forum itself describes their product as, "an open, global specification that empowers mobile users with wireless devices to easily access and interact with information and services instantly." WAP specifies a method to communicate across wireless networks quickly, efficiently, and securely. This communication can take place using mobile telephones, pagers, and personal digital assistants, and other mobile digital devices. Though the entire WAP specification is treated monolithically and has a single version number (the current version is 1.2), it is in fact a collection of specifications which define various layers and functionality of the protocol. The specifications draw heavily on existing Internet technologies (such as XML, URLs, scripting, and various content format) and mobile networking technologies (e.g. digital data networking standards).

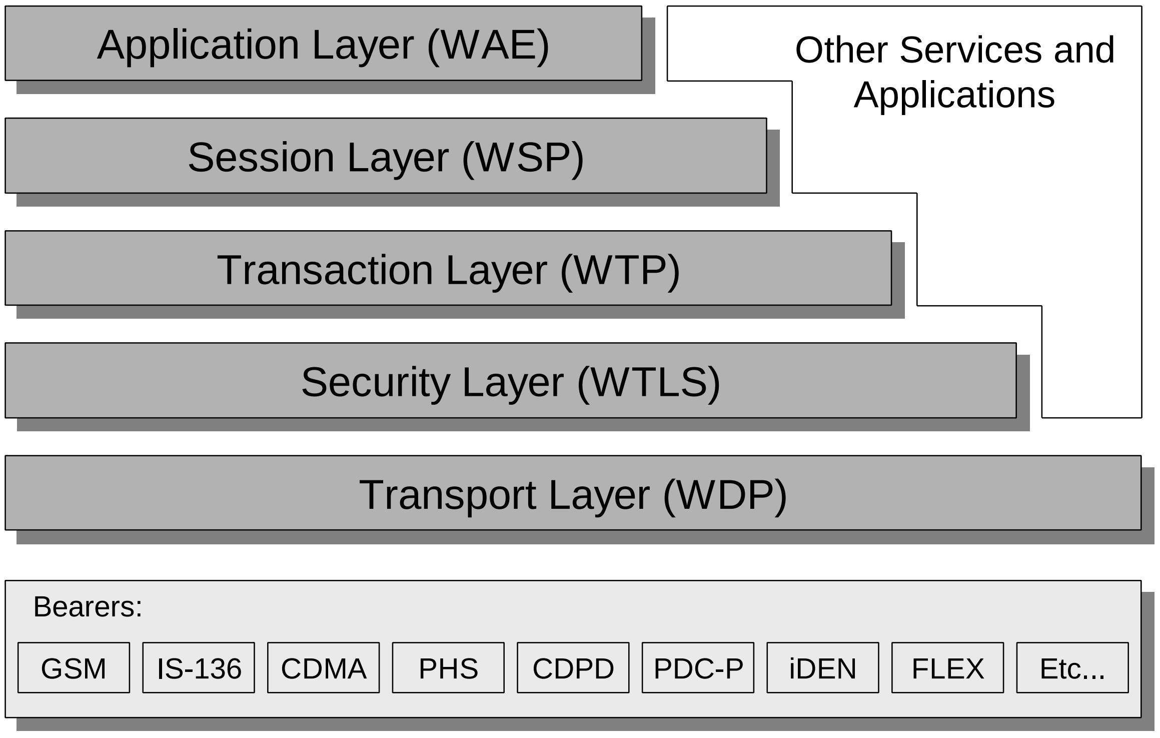

Figure 2: The Wireless Application Protocol architecture

WAP offers a number of features useful for wireless data communications:

- It provides a model that is very similar to the highly successful Internet and WWW, and is based upon ideas and standards in this domain. This approach reduces the time and effort required to develop WAP applications and resources.

- User display is the responsibility of a “microbrowser”, modeled on the familiar web browser. Apart from supporting WAP’s presentation/programming tools, Wireless Markup Language

(WML) and WMLScript, and the rest of the WAP protocol stack, details of the microbrowser are left unspecified. This leaves room for add-on innovations, just as Java, plug-ins, and other features have appeared in many web browsers.

- WAP is sensitive to the lower bandwidths, higher latencies, and reduced reliability of wide-area wireless connectivity. Binary encoding and other techniques are introduced at multiple levels in the WAP stack to reduce the volume of data transmission and to accommodate multiple simultaneous requests. For example, the WAP analog of HTTP uses binary coding of headers; WML and WMLScript are reduced to bytecodes prior to transmission.

- WAP makes minimal demands on the underlying operating system (OS) and is therefore compatible with virtually any available OS, ranging from the minimal embedded OSes used in pagers and cell phones to PalmOS and Windows CE on PDAs to desktop operating systems such as Windows, Unix, etc.

- WAP includes a Wireless Telephony Application (WTA) specification that is responsible for the actual interfacing of the device to the wireless network. This includes the handling of network events that allow the device to interact with a remote application.

In short, WAP is positioned at the convergence of two rapidly evolving network technologies, wireless data and the Internet. Both the wireless data market and Internet markets are growing very fast, which has fuelled the creation of a plethora of new information services.

3.2 Wireless Application Environment (WAE)

The Wireless Application Environment (WAE) Specification covers the user-level application framework for wireless devices such as mobile telephones, pagers and PDAs. It is based primarily on World Wide Web (WWW) technologies and philosophies [WAE], but targeting wireless connectivity and handheld devices impose important constraints on the design of the Wireless Application Environment. Web/Internet technologies have traditionally been based on relatively powerful desktops and workstation with high bandwidth and reliable/secure network. On the other hand mass-market handheld devices tend to have:

- Less powerful CPUs,

- Less memory,

- Restricted power consumption,

- Smaller displays, and

- Limited input devices, such as a phone keypad.

Wireless data networks tend to have:

- Less bandwidth,

- More latency,
- Less connection stability, and
- Less predictable availability.

The primary objective of WAE is to provide an interoperable environment for building applications that work across a wide variety of different wireless platforms. They also have to be reliable and secure. The specifications comprising the Wireless Application Environment provide this framework.

3.2.1 Wireless Markup Language (WML)

WML is a markup language built specifically for communicating across WAP-based networks, and is a specific instance of XML (eXtensible Markup Language). It is similar in purpose and appearance to the web's HTML, but is more limited due to the constraints mentioned above. The major functions of WML are:

- Text presentation and layout - WML supports text and image, including variety of formatting and layout command. (e.g. boldface text);
- Deck and Card organization metaphor - all information in WML is organized into collection of cards and decks. Cards specify one or more units of user interaction (such as a choice menu, a screen of text or a text entry field). Logically, a user navigates through a WML card, reviews the content of each, enters requested information, makes required choices and moves to another card. Cards are grouped into decks, which are similar to HTML pages in that they are identified by unique URLs and are the unit of content transmission over the network.
- Inter-card navigation and linking - WML supports navigation between cards and decks. Also WML handles events within the device, which may used for navigational purposes or to execute scripts. WML also supports anchored links, similar to those in HTML.
- String parameterization and state management - all WML decks can be parameterized using state model [WML]. Variables can be used in the place of strings and are substituted at run-time with their current value. It means that textual information of a card or attribute values in WML element (such as %vdata and %HREF) can be substituted at the run time. Substitution does not affect the current value of the variable and is defined as a string substitution operation. (e.g. This is a $var)

As in all XML schemas, WML elements describe markup information about a deck. An element may be of two forms: <tag>something</tag> or <tag/>. Most WML elements have attributes that let you specify additional information about how a WAP device should interpret an element. All attribute values must be enclosed within " or ' quotation marks. [XML] WML includes elements familiar from HTML: <p>, <a>, <table>, <tr>, <td>, <br>, in addition to the following:

WML Element	Description
<wml>	Defines a deck and encloses all information and cards in the deck
<card>	Defines text and input elements that support interaction with user
<input>	Prompts user to enter text string or number. Attributes of this element can limit entry numbers, set a maximum input length, or specify default input value
<do>	Declare action that user agent will perform when user presses the key (or button depend on device) specifies by the type attributes (e.g. go, pre,or noop). Label attribute specifies label to display for function key
<go>	Specifies URL to navigate to. The <go> element is used to specify task to perform <do> action. An attribute of this element defines HTTP submission method (e.g. get or post) to be used
<select>	Prompts user to choose one or more items from specified list. Attrinutes of this element specify single or multiple selection, title of element and default item selection
<option>	Specifies single choice in select element. An attribute specifies URI to navigate to when option is selected.
<img>	Include image (WBMP image: 1 bit version of bitmap image and should not exceed 150 X 150 pixel) in card. Attribute specifies URI for image, alternative text to display if images are not supported.
<postfield>	Specifies name-value pair to be transmitted to server during URI request. Attributes specify field name, field value.

The following is a simple example of a WML card deck. After loading the deck, the user agent displays the first card. If the user activates the DO element, the user agent displays the second card.

```
<wml>                                   // WML element, which defines a deck
<card>                                  // CARD element,
<p>                                     // Paragraph element
<do type="accept">              // DO element, of which pre-defined type is acceptance.
<go href="#card2"/>             // GO, which links to card2
</do>
Hello world!                    // text contents
This is the first card...
</p>
</card>
```

```
<card id="card2">              // CARD element, which is identified "card2"
<p>
Hello world!
This is the first card...
</p>
</card>
</wml>
```

It is important to recognize that WML is a specific XML schema with a well-defined DTD (document type definition). This means that it can be used as a presentation format for general XML documents, just as in the web world, XML documents are typically translated to HTML for presentation via web browsers. It also means that the traditional HTML-based pages still common on the WWW must be translated to WML before they can be displayed on a WAP-enabled device. While this may appear to be a major constraint, it must also be recognized that the limitations of WAP device displays will undoubtedly require some form of "adaptation" of arbitrary WWW HTML pages in order to present them in a readable form. Some of this issues may be addressed by web developers at the time they design their sites by giving due consideration to the variety of devices expected to access the site. [DEVSHED]

3.2.2 Scripting Language (WMLScript)

WMLScript is another part of the WAP application layer and it can be used to add basic client side programmability. [WMLSCRIPT] It can be used to check the validity of user input, features of the device. (e.g. initiating phone calls, sending messages, etc), and to produce error message and dialogs for the user. The fact that WMLScript is a client-side tool means it can play an important role in helping developers reduce the need for remote communication.

The WMLScript language is based on ECMAScript, which is a standardized derivative of JavaScript. But, unlike ECMAScript, the WMLScript specification also defines bytecode and interpreter reference architecture for optimal utilization of current narrowband communications channels and handheld device memory requirements. The following summarizes the basic syntax of WMLScript:

- The smallest unit of execution in WMLScript is a statement and each statement must end with a semicolon (;)
- WMLScript is case-sensitive
- Comments can either be single-line (beginning with //) or multi-line (bracketed by /* and */). This syntax is identical to both C++ and Java
- A literal character string is defined as any sequence of zero or more characters enclosed within double (") or single (‘) quotes
- Boolean literal values correspond to true and false
- New variables are declared using the `var` keyword (i.e. `var x;`)

- WMLScript is a weakly typed language. This means that no type checking is done at compile- or run-time and no variable types are explicitly declared. Internally, the following data types are supported: boolean, integer, floating-point, string, invalid. WMLScript will automatically convert between the different types as needed. Since WMLScript is not object-oriented (as Java or C++ are), it is impossible to create your own user-defined data types programmatically.

- WMLScript supports a variety of operators that support assignment, arithmetic, logical, string, comparison, and array operations. The set of supported operators is virtually identical to that of JavaScript.

- WMLScript supports a number of control statements for handling branching within programs. These include the if-else, for loop, while loop, break, and continue statements.

- Functions can be created and used in WMLScript, either internal to a given compilation unit or external to it. A function declaration has the following syntax:

  ```
  extern function identifier(FormatParameterList) Block ;
  ```

  The `extern` keyword is optional and is used to specify a function that can be called from outside the current compilation unit in which the function is defined. A sample WMLScript function declaration looks like this:

  ```
  function RunTime(distance, speed) {

  var time = distance / speed;

  return time; };
  ```

  The example takes two input variables, `distance` and `speed`, and uses them to calculate a time variable. The return keyword is then used to return a value.

- While WMLScript does not support the creation of new objects, it does provide six "pre-built" libraries that aid in the handling of many common tasks; other libraries are likely to be defined in the future, as the need for them is demonstrated. The current libraries are

  - Lang - contains a set of functions that are closely related to the WMLScript language core. Included in this library are functions for data type manipulation, absolute value calculations, and random number generation.

  - Float - is optional and is only supported on those clients who have floating-point capabilities. Typical functions provided by this library include `sqrt()`, `round()`, and `pow()`.

  - String - contains a set of functions for performing string operations. Some of the functions included in this library are `length()`, `charAt()`, `find()`, `replace()`, and `trim()`.

- URL - contains a set of functions for handling both absolute URLs and relative URLs. Typical functions include `getPath()`, `getReferer()`, and `getHost()`.

- WMLBrowser - contains functions by which WMLScript can access the associated WML context. These functions must not have any side effects and must return invalid in cases where the system does not support WMLBrowser and where the WML browser does not invoke the interpreter. Commonly used functions in this library include `go()`, `prev()`, `next()`, `getCurrentCard()`, and `refresh()`.

- Dialogs - contains a set of typical user interface functions including `prompt()`, `confirm()`, and `alert()`.

When calling a function included with one of the WMLScript standard libraries, the library name must prefix the function name. For example, to call the String library's `length()` function, use the following syntax:

```
var a = String.length("1234567890");
```

3.2.3 The User Agent

Although WAE does not formally specify any particular User Agent, the most common type of User Agent in the WAP architecture is a browser, which interprets WML and WMLScript. The WML user agent is a fundamental user agent of the WAE. However, WAE is not limited to a WML user agent. WAE allows the integration of domain-specific user agents with varying architectures and environments.

Because of the diversity of WAP-enabled devices, it is important that the content producer (or transformation gateway) be provided with information as to the capabilities and user preferences for the particular device being used. For this purpose, WAP has adopted the World Wide Web Consortium's (W3C's) Composite Capabilities/Preference Profile (CC/PP) approach, which uses the Resource Description Framework (RDF) to define a high-level structured framework for describing this information. CC/PP profiles are structured into named "components", each containing a collection of attribute-value pairs, or properties. Default values for each attribute are associated with the particular device, but can be overridden by explicit specifications. The User Agent Profiles (UAProf) specification extends WAP 1.1 to enable the end-to-end flow of a User Agent Profile, also referred to as Capability and Preference Information (CPI), between the WAP client, the intermediate network point, and origin server. It uses CC/PP model to define robust, extensible framework for transmitting and describing CPI for the client, user, and network.

The specification defines a set of components and attributes that WAP-enabled devices may convey within the CPI. This CPI may include hardware characteristics (screen size, color capabilities, image capabilities, manufacturer, etc.), software characteristics (operating System vender and version, list of audio and video encoders, etc), application/user preference (browser manufacturer and version, markup languages and version supported, scripting languages supported, etc.), WAP characteristics (WMLScript libraries, WAP version, WML deck size, etc), and network characteristics (bearer characteristics such as latency and reliability, etc.) [UAPROF]

3.3 Communications Protocols

As mentioned above, WAP is a complete protocol stack, with WML, WMLScript, and the user agent as the aspects most visible to the user. Beneath this are the communications protocols WAP specifies in order to convey the information between the source and the user's device. These aspects of WAP have less impact on its potential uses in handheld supercomputing, but do impact the operation and capabilities of a WAP device, and so are described briefly here.

3.3.1. Wireless Session Protocol (WSP) -- Session Layer

The Wireless Session Protocol (WSP) provides means for organized exchange of content between cooperating client/ server applications. It provides the means to:

- Establish a reliable session from client to server and release that session an orderly manner
- Agree on a common level of protocol functionality using capability negotiation
- Exchange content between client and server using a compact binary encoding
- Suspend and resume the session

WSP provides the application layer of WAP with a consistent interface for two session services. The first is a connection-oriented service that operates above the transaction layer protocols WTP (Wireless Transaction Protocol). The second is a connectionless service that operates above a secure or non-secure datagram service WDP (wireless datagram protocol), which is used if reliable delivery and confirmation are not necessary. [WSP]

WSP currently consists of services suited for browsing application (WSP/B), providing:

- HTTP/1.1 functionality -- All the methods defined by HTTP/1.1 are supported. In addition to this, capability negotiation can be used to agree on a set of extended request methods.
- Exchange client and server session headers -- HTTP/1.1 content headers are used to define content type, character set encoding. But compact binary encoding is defined for the well-known headers to reduce protocol overhead. Like MIME (multipart/mixed) format in HTTP/1.1, WSP also defines specific composite data format that provides content header for each component within the composite data object.
- Session suspend and resume with session migration -- the life cycle of a WSP session is not tied to the underlying transport. A session may be idled to free up network resources or save device battery. A lightweight session reestablishment protocol allows the session to be resumed with minimal overhead. A session may be resumed over a different bearer network (such as GSM, CDMA, etc.)

WAP offers the equivalent of HTTP's "basic" authentication (which goes by the same name), allowing basic password protection of portions of the WAP environment. The caveats about this form of authentication are the same as those for the WWW version as well: minimal encryption is used on the transmitted password, making it potentially vulnerable to interception. However this is partly offset by the presence of the Wireless Transport Security Layer described below.

3.3.2. Wireless Transaction Protocol (WTP) -- Transport Layer

A transaction protocol is defined to provide the services necessary for interactive "browsing" (request/response) applications. During a browsing session, the client requests information from server, which may be fixed point or mobile, and the server responds with the information.

WTP runs on the top of a datagram service and optionally a security service. WTP has been defined as a lightweight transaction-oriented protocol that is suitable for implementation in "thin" clients (mobile stations) and operates efficiently over wireless datagram networks. Features of WTP include:

- Three classes of transaction service:
    - Unreliable messages are sent without acknowledgement
    - Reliable messages are sent without acknowledgement
    - Reliable messages are acknowledged with exactly one reliable response
- No explicit connection set up or tears down phases. Explicit connection open and/or close imposes excessive overhead on the communication link
- Optional user-to-user reliability - WTP user confirms every received message
- Optional out-of-band data on acknowledgement to reduce the number of messages sent
- Message Oriented: The basic unit of interchange is an entire message and not a stream of bytes
- Asynchronous transactions: the responder sends back the result as the data becomes available [WTP]

3.3.3. Wireless Transport Layer Security (WTLS)

WTLS is a security protocol based upon the industry-standard Transport Layer Security (TLS) Protocol, which is based, in turn, on Netscape's Secure Socket Layer (SSL). In addition, WTSL has new features like datagram support, optimized handshake and dynamic key refreshing. WTSL has been optimized for the use over narrow-band communication channels and operates over the transport protocol layer. Applications are able to selectively enable or disable WTSL features depending on their security requirements and the characteristics of the underlying network (e.g. privacy may be disabled on networks already providing this service at a lower layer). The primary goals of the WTLS are:

- Data Integrity - WTSL contains facilities to ensure that data sent between the terminal and an application server is unchanged and uncorrupted

- Privacy - data transmitted between the terminal and an application server is private and cannot be understood by any intermediate parties that may have intercepted the data stream

- Authenticity of the terminal and application server

- Denial-of-service protection - detecting and rejecting data that is replayed or not successfully verified. WTSL makes typical denial-of-service attacks harder to accomplish and protects the upper protocol layers [WTSL]

3.3.4. Wireless Datagram Protocol (WDP)

The transport layer protocol in the WSP architecture consists of the Wireless Transaction Protocol (WTP) and the Wireless Datagram Protocol (WDP). The WDP layer operates above the data capable bearer services supported by the various network types. As a general datagram service, WDP offers a consistent service to the upper layer protocol (Security, Transaction and Session) of WAP and communicate transparently over one of the available bearer services.

Since the WDP protocols provide a common interface to the upper layer protocols, they are able to function independently of the underlying wireless network. This is accomplished by adapting the transport layer to specific feature of underlying bearer.

WDP supports several simultaneous communication instances from a higher layer over a single underlying WDP bearer service. The port number identifies the higher layer entity above WDP. This may be another protocol layer such as WTP or the WSP or an application such as e-mail. This reuse helps provide efficient operation within the limited resources of a mobile device. [WDP]

3.3.5. Bearers

The WAP protocols are designed to operate over a variety of different bearers services, including short message, circuit-switched data, and packet data. The bearers offer differing levels of quality of service with respect to throughput, error rate, and delays. The WAP protocols are designed to compensate for or tolerate these varying levels of service.

Since the WDP layer provides the interface between the bearer service and the rest of WAP stack, the WDP specification lists the bearers that are supported and techniques used to allow WAP protocols to run over each bearer. [WAE]

4 Comparison of WAE with Other Technologies

4.1 WML, HTML, and XML

XML (eXtensible Markup Language) has been developed by the W3C as a general approach to the description (markup) of information (primarily) for publication on the World Wide Web. It allows the user to define a set of tags via a document type definition (DTD). WML is a specific instance of XML, which has as well-defined DTD (http://www.wapforum.org/DTD/wml_1.1.xml). HTML is the predecessor of XML and is presently the dominant markup language on the WWW. XML and HTML both incorporate a document object model (DOM) and the "style sheet" concept to help separate content from presentation. The style sheet language (extensible style language, XSL for XML; cascading style sheets, CSS for HTML) allows presentational styles to be defined for components of the document defined by the DOM. Current practice in the WWW domain is that for purposes of presentation, XML documents are translated into HTML (possibly combined with CSS) on the server side via XSL. As XML-aware browsers gain market share it may be increasingly possible to XML+XSL directly, but in practice content providers will probably have to provide server-side translation to HTML for some time to come.

As an XML instance, WML could in principle be treated in a similar fashion – translation via XSL into other formats or schemas. However in the context of WAP, WML is meant to be a presentation format more than a source format. If a mobile phone or other communications device is said to be WAP-capable, this means that includes a microbrowser that fully understands how to handle all entities in the WML 1.1 DTD and is not required to support more generic XML and XSL. It is important to note that WAP defines neither a document object model for WML nor and equivalent to XSL. Just as HTML is used as the presentation format on the WWW, arbitrary XML can in principle be translated into WML for presentation on a WAP via the appropriate XSL style sheet. In this sense, WAP leverages current Internet technologies and provides a straightforward path to allow content providers to support both traditional WWW browsers and WAP devices if they wish.

Recognizing the limitations of handheld devices, WML has limited functionality compared to HTML in many respects. Text formatting in WML document is limited to `<em>`, `<strong>`, `<i>`, `<b>`, `<u>`, `<big>`, and `<small>`. [WML] WML also does not support frames. However WML does provide for the use of string parameterization (in conjunction with WMLScript, see below) which is not part of HTML. Besides the specific constraints of the language, WAP content developers must also conform to other limitations inherent in the environment:

- Display Size – smaller screen size and resolution, a small mobile device such as a phone may only have a few lines (2-10 lines) of textual display, each line containing 8-12 characters. Palm series has bigger screen. But they also have very limited size from 160 X 100 pixels up to 320 by 320 pixel. It is very limited capability, comparing with 20", 1028 X 768 pixels of general purpose PC screen.

- Input device – a limited, or special-purpose input device. A phone typically has a numeric keypad and a few additional function-specific keys. Input is usually accomplished with combination of a numeric keypad and a function key. Palm-size device supports a pointing device, hand writing recognition, etc.

- Computational resources – low power CPU and small memory size; often limited by power constraints.
- Narrowband network connectivity - low bandwidth and high latency. Devices with 30bps to 10kbps network connections and 5-10 second round-trip latency are not uncommon. [WML]

Many aspects of the particular device can be conveyed to the content provider using WAP's User Agent Profile mechanism, but the content provider must still accommodate them in an appropriate fashion. Two particular aspects that will require attention are the display size and capabilities and the size limits on individual cards and card decks which the unit can receive. The default limit on individual cards, for example, is 4096 bytes (though individual devices may have higher or lower limits).

4.2 WMLScript, JavaScript, ECMAScript and Java

JavaScript, ECMAScript, and WMLScript are all primarily designed to provide client-side programmability for use by content developers (Jscript, Microsoft's client-side scripting language, is essentially JavaScript with differences introduced for competetive as much as technical reasons; here it will be lumped with JavaScript). JavaScript is the most familiar of the three. ECMAScript is a derivative of JavaScript that has now been through the formal ISO standardization process. Among the most significant differences between JavaScript and ECMAScript is that the document object model which was part of JavaScript was separated from ECMAScript in order to allow the W3C to standardize a DOM. This means that JavaScript programs have the ability to access and manipulate parts of the document of which they are a part, this capability has been removed from ECMAScript. WMLScript is based on the core of the ECMAScript language, but designed recognizing the limited capabilities of WAP devices. Important differences between WMLScript and its predecessors include:

- Because it is derived from ECMAScript, WMLScript does not include the ability to directly manipulate the document to any great extent. WML can be parameterized using variables; WMLScript can manipulate those variables and force a refresh of the display. However since these variables can only be used for WML content and attributes, not tags themselves, there is a limit to this technique.
- Basic DOM functions for global navigation history and browser-control were swept into WML instead of WMLScript. A very limited event model is also part of WML.
- Server Tool - JavaScript has a server tool from vendors (including Netscape and Microsoft); WMLScript is a client-only scripting platform.
- For optimal utilization of current narrow-band communications channels and device memory requirements, WMLScript is compiled to bytecode before transmission to the client, and the bytecode is interpreted on the client side. There is nothing to prevent utilization of intermediate bytecode in implementations of JavaScript or ECMAScript, but at present there

is no mechanism in the WWW domain to distinguish byte-compiled scripts from the text source which is currently used.

- WMLScript is weakly typed (types are supported internally, but not specifically defined or checked in scripts.
- Support for the floating-point type is optional in WAP
- The WAP Forum bends URI semantics to allow special fragment-identifiers to represent WMLScript function calls instead of Web addresses.

JavaScript, ECMAScript, and WMLScript are scripting languages of modest size, designed primarily for client-side programmability and manipulation of content. Java, on the other hand, is a high-level general purpose programming language that is capable of supporting complex and sophisticated programs. It happens to have a variety of features that make it particularly amenable to network programming, but it is not limited to this domain. Although in principle, Java can be compiled to a native executable for a particular processor, it is usually implemented in a virtual machine environment. The Java is compiled to a bytecode which is then executed by a well-defined virtual machine (VM) which provides for both portability and security (because the VM limits operations which might be considered insecure in a network programming situation and gives the person executing the program the choice of granting the program additional privileges if they are requested). The Java VM implementation can be rather large, and present Java implementations rely on an extensive set of standard classes which are also rather large, even when compiled to bytecode. These requirements place Java beyond the reach of many WAP-enabled devices, however it is possible to run Java on some of the relatively high-end WAP devices, such as PDAs. There is a Java implementation already available for some devices running the EPOC32 operation system (Symbian is the primary manufacturer of such devices) and WABA from WabaSoft Inc. is a limited subset of Java designed for portable devices. Finally, Sun Microsystems is developing a small memory footprint “K virtual machine” (KVM) specifically targeted for mobile devices (including cell phones) and network appliances. WAP 1.1 does not include Java, but neither does it prevent someone from producing a WAP-compliant system that also supports Java. It is clear that these small-footprint Java implementations will have to sacrifice some functionality to achieve the memory savings, however it is not yet clear if or when those working in this area will converge on a particular model. Until they do, it is quite hard to gauge how useful Java will be in handheld supercomputing.

4.3 Comparison of WAP Protocols with other Technologies

Fundamentally, the Wireless Session Protocol (WSP) is the binary equivalent of HTTP/1.1. This means that all features of HTTP are supported by WSP. But since WSP is designed to operate over a narrow band communication network, WSP also includes a number of features not found in HTTP/1.1: [WSP]

- In addition to the usual HTTP “pull” mechanism, WSP provides a mechanism to allow the server to push data to the client asynchronously. Three types of transfers are supported:

    - Confirmed data push within an existing session context (Push data to client any time in session. Server will receive confirmation that the push was delivered)

    - Unconfirmed data push within an existing session context (as above but without confirmation)

    - Unconfirmed data push without an existing session (A default session context is assumed. This can be used to send a one-way message over unreliable transport)

- It is also possible to negotiate support for multiple, simultaneous asynchronous transactions – the client can submit multiple requests to the server. This improves utilization of airtime in that multiple requests and replies can be coalesced into fewer messages. This also improves latency as the result of each request can be sent to the client when it becomes available.

4.4 Comparison of WAP with Current Approaches to Mobile Web Access

The popularity and tremendous growth of the World Wide Web make it an obvious source for information services of interest to mobile users. Though WAP is distinct from the WWW and formally independent of it, WAP is clearly modeled on the web domain and it is logical to consider how WAP might be used to access existing WWW resources as opposed to specialized resources developed specifically for use by WAP-enabled mobile devices.

At present, there are essentially two approaches to providing mobile web access. One is direct browsing of URLs using a traditional web browser or a WAP browser. The other is “web clipping” in which the mobile device has a specific, programmatic interaction with web sites that typically selects specific information or components from the WWW version of the site to provide to the mobile user. (We must acknowledge that probably the majority of web clipping applications involve interactions with WWW sites that have already been tailored for mobile access rather than a general site which someone using a WWW browser on their desktop computer might also use.) In both approaches, communications may take place in real time or the information may be provided for “off-line” use, where the necessary data is downloaded in advance and can be viewed at any time by the user.

Clearly, any device hoping to provide direct access to standard WWW pages must have sufficient computational power to run a traditional web browser and a display on which a reasonable approximation of the web page can be rendered. This limits direct web access to relatively “high-end” handheld devices such as PDAs. Web clipping applications may be more amenable to use on more limited devices, such as cell phones.

For various reasons, direct web access is more readily available at present on PDAs running Windows CE than on other platforms. Microsoft’s Internet Explorer is often available on these platforms, as well as a variety of other products: Pocket Browser from Conduits.cm, lacks Java support, but does handle all HTML and image content [CONDUITS]; Spyglass Device Mosaic 3.2 supports Jscript, HTML, HTTP, and SSL [SPY]; PocketIE 3.0 from Microsoft supports frames, .WAV files, Jscript, and 128bit encryption. [POCIE] On the PalmOS platform (particularly the Palm VII, which incorporates a CDPD modem internally) SnakeEyes1.2 from snakefeet.com [SNAKE] provides access to arbitrary URLs by stripping out all of the HTML markup and returning just clear text. To help manage slow/expensive network usage, SnakeEyes gives you control of how much data sites should send. One of the significant problems at present with direct web access from wireless devices is that the bandwidth is much lower than on most desktop systems – often just 9600 bps! For this reason off-line browsing may be a more satisfactory approach if viewing of complete "native" WWW pages is a requirement.

SnakeEyes is an example of an approach that lies somewhere between direct web access and web clipping. This approach involves filtering and transforming data to a form more suitable for presentation on the mobile device. Plucker, from GNU design group is a suite of programs, which provides an off-line web browser for PalmOS systems with a similar transformation gateway on the PDA's host desktop system. [PGNU] AvantGo [AVANTGO] and Pendragon [PENDRAGON] provide more limited off-line browsers on PalmOS platforms. Both of these vendors provide access to a limited collection of web sites. While not stated explicitly, it is likely that the web pages offered through these systems have already been tailored to the Palm platform by the content providers.

The Palm VII provides the primary example of the web clipping approach. "Palm Query Applications" (PQAs) are associated with specific information resources (i.e. web sites) and utilize a local template to minimize communications requirements. When the PQA is invoked, the appropriate data query is sent to the web site. The returned information is fit into the template and displayed for the user. In principle, the model is capable of supporting direct web browsing, though communications costs make this prohibitive. In practice, the web component of the PQA is most conveniently (from the standpoint of a PQA developer) provided as specialized web pages specifically designed for use with the PQA.

5 Where Does WAP Fit In?

In the preceding sections, we tried to present a simple description of WAP and a comparison with related technologies that are more familiar from the World Wide Web domain. In this section we draw on those comparisons to try to understand how we should view as a tool for conveying information and for building information and computing resources, as is now being done in the WWW. While in the preceding sections we have tried to be as objective as possible, this section is necessarily colored by our perceptions, our experience and to some extent by the approach we have taken in the past to the development of web-based resources. We do not necessarily expect everyone to come to precisely the same conclusions as we do, and since WAP is at the confluence of several rapidly moving fields, we expect the picture will change with time.

5.1 The Web in the Palm of Your Hand?

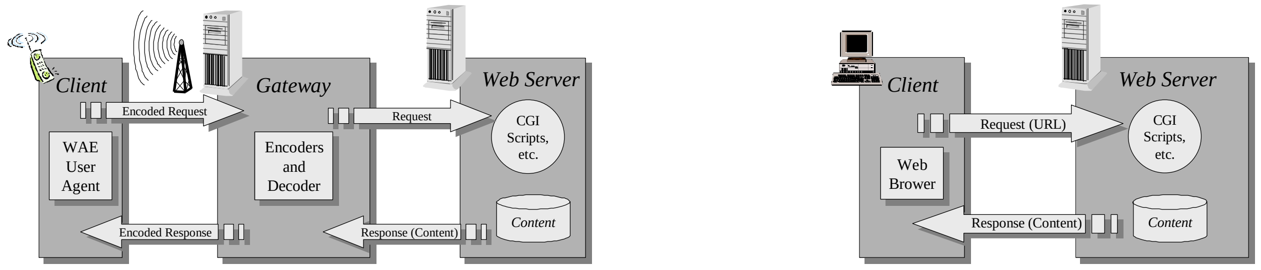

Figure 3: WAP (left) and WWW (right) programming models

The architecture of WAP is intentionally quite close to that of the World Wide Web in terms of data formats, protocols, and nomenclature (URLs, etc.). While WAP makes no promises about being able to provide access to arbitrary WWW resources (as opposed for those specifically designed for use with WAP devices), it is logical to consider what the prospects are for this. This question can be considered both from the point of view of the content provider/developer, and the end-user who wants to access existing WWW resources.

Despite the original “write once, present anywhere” philosophy of the WWW and increasing attempts to support the separation of content from presentation (i.e. stylesheets, the introduction of XML, etc.) the fact is that a great deal of current web content is “highly produced” and does not provide the desired separation. Taking this approach typically forces content developers to provide multiple versions of each web page, laid out for presentation of specific devices (combinations of browsers and platforms in the WWW domain). In this case, WAP is yet another display device, and in fact probably different versions would be required for specific devices (i.e. cell phone vs PDA, PDAs with different display sizes, color vs monochrome, etc.). The disadvantage is that each different display requires effort to develop and support, so content providers will not support devices they believe require too much effort relative to the user base they can reach. Because WAP devices are generally quite limited relative to traditional WWW displays, it will require more work to support them with content of comparable quality.

Content providers who use XML as their base data format and make use of the associated tools and machinery to translate the content for presentation on browsers are likely to have an easier time incorporating WAP into their repertoire. However if a single XML source is used to produce sensible presentations on both a 1024x768 pixel WWW browser and a 160x160 pixel WAP microbrowser (or even a cell phone display!) a great deal of thought is going to have to be given to the original content and the format translation. Given the diversity of WAP devices, and the disparity compared to WWW browsers, it is not at all obvious to us that a single-source approach can be made to work in a fashion that will be accepted by the end user. Another version of the single-source approach involves the introduction of a translation gateway (or WWW-WAP proxy server) to convert web content for WAP displays. In this case, the gateway is likely to be offered by a third party rather than the original content provider, or perhaps the wireless Internet Service Provider (WISP) providing connectivity to the WAP device. A number of companies are developing the technologies for such gateways, but as with single-source XML, it remains to be seen if they can provide satisfactory service to the end user. Because they must be more general than XML translation technologies, they face greater challenges. But this approach seems to be the only one that would allow users to access arbitrary web resources from their mobile devices.

The end user, of course, does not care too much about the technologies behind delivering WWW content to their display device, however they will (understandably) expect them to work. Given the obstacles content providers must surmount to provide WAP-accessible content on an equal footing with that for traditional WWW displays, we can expect that any service advertised as providing general WWW access via a WAP device will disappoint the end user. If, on the other hand, users are lead to expect access to specific content (tailored behind the scenes to WAP), we can expect a more positive response. There is also a question as to how the end user will wish to use their WAP devices. If users expect their WAP device to be interchangeable with their desktop WWW browser, then access to arbitrary web resources will be given more attention by content providers. If they expect to use WAP devices for specific tasks while on the move, and their desktop WWW access for general browsing, then the emphasis will be on the development of specific resources of value to mobile users.

Given the current situation, with wireless connectivity relatively expensive and performance poor, combined with the significant differences between typical WWW displays and WAP displays, we can expect that most WAP users will utilize online resources that are tailored to mobile devices and that delivery of arbitrary WWW content to WAP devices will be a much-discussed (and much-hyped) area of development. With increasing wireless connectivity (perhaps including broadband wireless in metropolitan areas), increasing use of XML and related machinery as the primary data format on the web, and/or the development of robust WWW-WAP proxy services, WAP devices will become increasingly integrated with the rest of the WWW community.

5.2 WAP on the Desktop?

The question of whether WAP will be used on the desktop is motivated by the fact that today's browser technologies are becoming increasingly bloated and resource hungry, as well harboring many bugs. As a simpler, well-defined technology, one can imagine small high-quality WAP microbrowsers implemented for traditional desktop operating systems providing an alternative to the general WWW browser. Such browsers are already under development to assist WAP content developers:

- WAPMan – Windows 9x/NT, from VirtuaCom.com
- Opera – Windows 9x/NT, Linux, from Opera.com
- WINWAP – Windows 9x/NT, from SlobTrot.com

On the positive side, we expect it will be relatively easy to produce high-quality microbrowsers due to the limited functionality. Because these browsers, and the content they're designed to display are typically targeted at small display sizes, this may prove to be an interesting option to those who wish to have instant access to a variety of information "channels" on their desktop, but don't want to devote a tremendous amount of screen real estate or workstation resources to them. In this sense, WAP browsers may provide a substitute for "news tickers", though it is not clear if they offer any real advantages.

On the negative side, WAP is a very limited standard, and a WAP-compliant browser is not required to provide many of the services users have come to expect in desktop WWW browsers: Java, e-mail client, news client, ftp client, etc. These capabilities contribute significantly to the size of current WWW browsers, and if users demand them in desktop WAP browsers, they will face the same risk of bloat. The other problem with WAP on the desktop is the content. As discussed above, it seems likely that arbitrary access to WWW content from WAP displays is some time off.

5.3 Commercial Applications

Clearly, the acceptance and development of WAP-enabled resources will be driven by the market for commercial applications of this technology. Given the growth expected, particularly in the cell phone area, there are projections that the market for wireless information services will quickly exceed the number of traditional desktops with Internet access. Initially, these services will be targeted at the needs for specific types of information delivered "any time/any place", and location-aware services for mobile users. Examples include travel information, stock market data, and Yellow Pages style directory information. Next will come a new breed of "universal access" services that use WWW/WAP technologies to provide users with access from any WWW/WAP capable device. Web-based email access, browser bookmark repositories, and storage of arbitrary files (including, for example, PDA data backups) are already starting to appear, and the variety of services will expand.

One interesting question with respect to the development of commercial applications in the WAP domain is what will the business model be? Presently a great many WWW services are free to the user, supported by advertising. On the limited displays of WAP devices, traditional banner advertising becomes much more intrusive and may not prove acceptable to the end user. On some devices with audio capabilities (particularly cell phones), it may be possible to make that channel for advertising purposes, but there are technical problems, including size of audio data streams, and the fact that audio capabilities are not yet universal in prospective WAP devices. The types of services described above may have enough value to mobile users that they will be willing to pay for them on a per-use or subscription basis. Access to arbitrary WWW resources is more problematic, especially if a for-fee third-party gateway is translating content (including ads) from sites – perhaps denying the original provider access to the advertising "impressions" of mobile users.

It must also be noted that in the WWW domain at present, many of the most sophisticated applications depend heavily on client-side tools such as JavaScript or Java, or external plug-ins. Since WAP says nothing about the available of such tools (with the partial exception of WMLScript), and at present most WAP-capable devices do not offer such capabilities, commercial applications are going to have to rely primarily on server-side programming if they want to provide portable and easily accessible services.

5.4 WAP for HPC, Research, and Education

Use of WAP devices in high-performance computing, research and education is predicated on the commercial success of the technology. Further, "handheld supercomputing" benefits when it overlaps with particular applications or techniques that are popular and therefore well developed in the commercial arena. Perhaps the most straightforward concept to translate into the handheld supercomputing arena is the any time/any place information services. These can include status information about machines and batch queues, job status and progress tracking, and simple computational steering. We also expect simple forms of collaboration (i.e. chat/messaging applications) and basic multimedia applications (MP3 tools, etc.) to become popular on handheld devices, and these technologies will directly or with minor adaptation support collaboration in the research and education domains. Some of the infrastructure and actual applications of the "universal access" type will also be useful in the handheld supercomputing domain; others are special to the domain and will have to be developed.

In general, it appears that developers of handheld supercomputing applications should view WAP mainly as a display technology with very modest client-side programmability. This means that primary processing of information must be done on the server side, with the minimum information transmitted over the air to the client for display. Some developers may find this model rather limiting, and as described below, we expect to see a richer programming model for handheld devices emerge in the next few years, but limiting oneself to WAP is clearly the safest approach at the present time and has the advantage of a very high level of portability.

6 Looking Ahead

As Internet and telecommunications standards go, the Wireless Applications Protocol is relatively mature, with the second revision of the standard (version 1.2) now in use. Devices, software, and web resources supporting WAP are starting to reach the public and there seems to be great enthusiasm behind this movement. Though the trade press has reported grumblings from some content providers about the additional burden of supporting a multitude of WAP devices with different display characteristics, it seems to us that they must persevere. The alternatives would seem to be either a new standard or a balkanization of the wireless data market as vendors produce their own, incompatible solutions. Given current technologies, there is no reason to believe that a substantially better standard can be achieved, and fragmentation of the market would leave content providers in a tougher situation yet. With more powerful devices and better network connectivity, it might become feasible to shift from WAP to a more WWW-like model (i.e. use of HTML or even XML+XSL directly). However, since the display size issue is unlikely to be changed substantially by such advances, content developers will still have to face the most labor-intensive aspect of supporting mobile data communications with or without WAP.

The amount of content accessible via WAP devices will be governed by several factors. After the "cherry picking" of developing resources targeted specifically to mobile users, the primary issues will be the success of WAP-WWW proxy gateways offering automatic translation of arbitrary WWW sites into WAP, and the general uptake of XML and XSL as the underlying data format of the web. As we have repeatedly observed, the display size issues do not go away simply because a site uses XML, but certainly it increases the chances that people will be motivated to develop and deploy appropriate style sheets for WAP devices which can be reused by many sites.

Finally, we must note the effect that Moore's Law will undoubtedly have on handheld digital devices in the coming years. The rapid increase in performance of CPUs, memory density, and reductions in power consumption will allow future generations of handheld devices to be made more and more computationally sophisticated. We can also expect improvements in wireless connectivity, though perhaps not at the same pace. This will make it feasible to add features not commonly found on today's handhelds, such as Java implementations. Such improvements *may* provide a new level of portable client-side programmability in conjunction with a display technology like WAP. The qualifier is important, because of the problems we see now in the WWW domain:

- Vendors ignoring standards for competitive reasons
- Close coupling with web browsers inhibiting uptake of the latest Java standards and tools
- Substantial variations in Java performance across platforms, making it effectively unusable on some

These problems may be overcome if the WAP Forum acts quickly to incorporate Java into their standards in some fashion, but at this time and there is sufficient market interest in the capabilities it provides. However at present, the question of what is the appropriate "minimal Java" for resource-limited implementations remains open and will have to be resolved before Java can be usefully incorporated into WAP. We expect that Java will be particularly interesting for some handheld supercomputing applications, and there may be sufficient motivation to begin using it even in the absence of standardization.

Despite Moore's Law, however, the performance, capabilities, and connectivity of handheld digital devices must always be expected to lag behind those of desktop workstations. Consequently, something akin to WAP, which explicitly recognizes these differences, will always be a reasonable approach to the mobile data market. Therefore, we expect that WAP will evolve with time to reflect advances in both mobile and desktop technologies, though future standards may not always be called WAP.

References

[AVANTGO] "Mobile Internet", AvantGo, avantgo.com/developer/web/index.html

[CERTI] "The ECC Tutorials and Whitepapers", Certicom Corp., www.certicom.com

[CONDUITS] "Pocket Browser 1.5" conduits technologies inc., www.conduits.com/ce/browser/

[DEVSHED] "Introduction to Wireless Access Protocol", W.J. Gilmore, 18-Feb.-2000.

[DEVNET] " Adding Client-Side Logic To WAP Using WMLScript, Wireless Developer Network

[KVM] " K Virtual Machine", Sun Micro Systems, java.sun.com

[MOBEXPLORER] " A Modular Application Platform for Mobile Phones", Microsoft.com, http://www.microsoft.com/windowsce/wireless/MobileWhitepaper.rtf

[PALMNET] "Palm Computing Paltform: Development Zone", Palm, www.palm.net

[PENDRAGON] "Pendragon Inforover", Pendragon Software Corp., www.webfayre.com/inforover.html

[PGNU] "Plucker: FAQ", GNU-designs, plucker.gnu-designs.com/FAQ.shtml

[PHN01] "Enabling the Wireless Internet", Unwired Planet, Inc., Feb.-1999

[PHN02] "WAP Overview", Phone.com, Feb.-2000

[POCIE] "Pocket Internet Explorer", Microsoft, www.microsoft.com/windowsce/Products/hpc/pie.asp

[SMARTRAY] " What is smartRay.com?", smartRay.com, www.smartray.com

[SPY] "Spyglass Device Mosaic 3.2", SPYGLASS, http://www.spyglass.com/solutions/technologies/devicemosaic/

[UAPROF] " WAP User Agent Profile Specification" 10-Nov.-1999, WAPForum

[WABA] "Product overview", wabasoft, www.wabasoft.com

[WAE] "WAP Wireless Application Environment Overview", WAP Forum, 04-Nov.-1999

[WML] "Wireless Application Protocol Wireless Markup Language Specification ver 1.2", WAP Forum, 4-Nov.-1999.

[WDP] "WAP Wireless Datagram Protocol" 5-Nov.-1999, WAPForum

[WMLSCRIPT] "Wireless Application Protocol WMLScript Language Specification ver 1.1", WAP Forum, 4-Nov.-1999.

[WSP] "WAP Wireless Session Protocol Specification" ver 5, Nov.-1999, WAPForum

[WTP] "WAP Wireless Transaction Protocol" 11-Nov.-1999, WAPForum

[WTSL] "WAP Wireless Transport Layer Security" 5-Nov.-1999, WAPForum

[XML] "Wireless Markup Language" May-2000, XML Journal

Appendices

A WAP Today

A.1 Current Implementations and Projects

1. WAP is?

    http://www.phone.com/industry/wap.html

2. Phone.com

    http://www.phone.com/index.html

3. Windows CE has Pocket Internet Explorer

    http://www.microsoft.com/windowsce/products/hpc/network.asp

4. AvantGo on PalmPilot

    http://avantgo.com/

    http://corp.avantgo.com/info/datasheet/specs.html

    --> CodeWarrior http://www.metrowerks.com/desktop/wintools/

5.1 Palm Query Applications

    http://www.tow.com/software/pqa/

5.2 Web Clipping

    http://www.tow.com/publishing/Developing_Web_Clippings/index.html

6. Web clipping

    http://www.palm.net/

7. PalmPilot

    http://www.palmpilot.com/home.html

8. PalmTop

    http://www.hp.com/jornada/

9. Palm Software

www.palmgear.com

10. Palm Dev Spec

http://www.palm.com/devzone/docs/palmos/index.html

11. Nokia WAP

http://www.nokia.com/wap/index.html

12. Pendragon InfoRover

http://www.webfayre.com/inforover.html

13. WABA soft

http://www.wabasoft.com/

14. WAP FAQ

http://wap.colorline.no/wap-faq/

15. Synchronization Standard Org. SyncML

http://www.syncml.org/

16. Hardware and Software shop mypilot.com

http://www.mypilot.com/

17. WAP Protal Site

http://www.wapaw.com/

18. WML browser company from Asia/Pacific

http://www.thewirelessedge.com/

19. Wireless portal site and service

http://www.smartray.com/

20. Yahoo Directory

http://dir.yahoo.com/Science/Engineering/Electrical_Engineering/Telecommunications/Wireless/Wireless_Application_Protocol__WAP_/

21. Ericcson WAP page

http://www.ericsson.se/WAP/

22. WAP search engine

    http://www.wapmap.com/

A.2 WAP-Enabled Web Sites

1. TagTag http://www.tagtag.com Free WAP site hosting and online Java based WAP site editor

2. Freedom2Surf http://www.freedom2surf.net Free HTML service provider, but they support PHP3 which makes it ideal for providing WAP content

3. Anytimenow.com http://www.anytimenow.com Free provider that offers 20MB space, online editing and storage of `.wml` and `.wmls` files. It also allows e-mail access via WAP, including Hotmail.

4. WAPMAP http://www.wapmap.com/

5. WAPAW http://www.wapaw.com/

6. SmartRay http://www.smartRay.com

7. WAPmine http://www.wapmine.com

B Handheld Supercomputing Today

Computer-based collaboration tools are an emerging technology that promises to change the way researchers, educators, and others are able to interact with colleagues or students over long distances. Typical collaborative systems available now allow not only audio and video conferencing capabilities, but also the sharing of documents and joint editing, shared or guided web browsing, shared whiteboard, and numerous other capabilities.

In order to gain a better understanding of what it possible today in terms of “handheld supercomputing”, and how technologies like WAP might effect the situation, we developed a simple demonstration, linking a desktop-based collaboration system and a handheld personal digital assistant using “native” technologies. The collaboration system is Tango Interactive 2.0, developed at the Northeast Parallel Architectures Center at Syracuse University and commercialized by WebWisdom.com. Tango provides a shared event collaboration model using a client-server architecture. Client-side control is provided by the combination of the Tango browser plug-in and the Session Manager Java applet. The Session Manager provides the means to launch other collaboration clients locally, and, through the Tango Server, can cause other participating Session Managers to launch clients or simply provide information about the session, which they can join at their option. Individual clients are often Java applets as well, but can be native applications. Clients communicate with each other via the Tango Server, or may setup their own channels outside of Tango (this is done in the case of audio/video conferencing, for example, to minimize the load on the Server). The particular collaborative client chosen for the PDA linkage is a simple “chat” application that allows users to share text messages. Chat was chosen because it is relatively simple to implement on the PDA, and it represents most of the fundamental issues in linking handheld devices with fixed collaboration tools.

The PDA is the popular Palm III from Palm, Inc. (recently spun off from 3Com). For this experiment we used the add-on modem available for the Palm to make a PPP network connection because it was the simplest way to provide network connectivity (wireless data coverage was not available in the Syracuse area at the time this work was started, but Palm.net’s CDPD service for the Palm VII has since expanded into this region). The particulars of the network connectivity are not terribly important for the purposes of this demonstration. The Palm chat application was developed using the Metrowerks CodeWarrior integrated development environment (IDE). Despite the availability of a high-quality IDE, development of Palm software applications can be likened to the state of Windows programming 8-10 years ago: the programmer must handle many lower-level details that in the Windows world are now handled by pre-built widgets. For example, a typical widget available today would be a text area with scroll bars if the text is too large to fit in the display area. On the Palm, the programmer must “manually” add the scrollbars. Of course just has the Windows environment has evolved over time, we can expect the same to happen with the Palm. Though CodeWarrior includes a debugger, it is also useful to have the PC-based Palm emulator package (available from Palm, Inc.) which allows Palm code to be tested thoroughly without the need to actually download it to the physical device (though some things, such as network events, do not work in the emulator exactly as in the actual device).

In this work, we implemented an architecture very similar to that used in WAP: the desktop Tango chat application was modified to add a "Palm Proxy", which simply passes input from other clients along to the gateway to the Palm device, and accepts inputs from the Palm via the same gateway. The gateway, or "Palm Server" handles the intermittent connectivity of the handheld via a store-and-forward approach, and the Palm device connects over the network to the Palm Server rather than to the Tango system directly. The gateway could also be used to apply filters or transformations to the data passing through it in order to better accommodate the capabilities and performance of the PDA. For instance, if network connectivity is very expensive, a PDA user may wish to receive only specific chat messages (i.e. those containing the phrase "earnings report"). The introduction of a gateway also avoids the need for the Tango system to know anything about the use of the PDA, and the need to port the entire Tango protocol stack, Session Manager, and other machinery to the PDA. By introducing Palm Proxies on a per-client basis rather than a per-user basis (Session Manager level), we simplify the introduction of handheld devices because that way they can most easily be tailored specifically for the application and the PDA's capabilities.

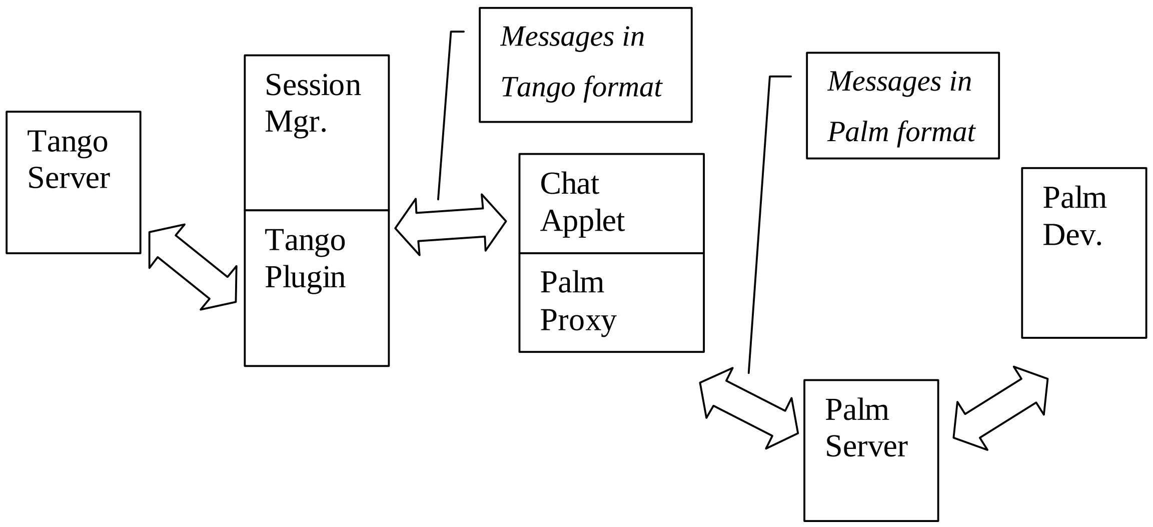

Figure B.1: Architecture of Tango-PDA system

It is worth noting that at present, the capabilities of the Palm are somewhat limited with respect to the development of complex collaboration tools and other applications. In particular the platform does not offer multiprocessing/multithreading capabilities. Other PDA operating systems (i.e. Windows CE and EPOC) may or may not provide better models in this respect, but in any case, we can expect that the capabilities of handheld operating systems will become more sophisticated with time.

Although WAP is not intended to provide an environment in which to develop sophisticated collaboration tools, it has, in principle, sufficient capabilities to at least approximate this particular text-based collaboration application. In a WAP implementation, the Palm Server would be replaced by a WAP Gateway. The Palm Proxy component of the desktop chat would provide the contents of the chat window as a WML card, and new inputs on the Tango side can be sent to the PDA using WAP's Push Architecture, which is capable of reliable delivery to the wireless client (analagous to the Palm Server's store-and-forward approach to reliable delivery). User input on the PDA can be accepted using WML's `<input>` and `<postfield>` tags. There may be timing and other issues in such an implementation (for example, what happens if the PDA received a push of new input from the Tango side while the PDA user is composing a text string to be sent to Tango), but it does appear that WAP provides for at least a rudimentary implementation of this text-based application with no programming on the PDA side.